DELTA CACHING SERVICE

RELATED APPLICATIONS

[0001] This application is related to:

U.S. Patent Application Serial No. 09/734,910, filed December 11, 2000, entitled "Predictive Pre-download Using Normalized Network Object Identifiers";

U.S. Patent Application Serial No. 60/263,247, filed January 22, 2001, entitled "Server Driven Differential Caching"; and

U.S. Patent Application Serial No. 10/058,232, filed October 19, 2001, entitled "Differential Caching with Many-to-One and One-to-Many Mapping", all of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The invention relates to a delta caching service, and related methods and systems.

Description of Related Art

[0003] In a system that delivers information from a server to clients who request that information (such as a web server delivering information to a set of web clients), it is desirable to minimize the amount of data that is actually sent from the web server to the web client. Delta caching is a technique by which the server and the client differentiate between template information and delta information for an object to be delivered from the server to the client; the client maintains a copy of the template information and the server is therefore able to deliver the object by only sending the delta information.

[0004] The web server often manages relatively large loads by dividing its tasks, using a load balancer, among a set of server-responders. However, differing server-responders might then associate different template information with objects at the server. One consequence is that when a client communicates with more than one server-responder, the template information known to the client and to the server-responder might differ. This is increasingly more likely as the number of server-responders is increased, and might result in miscommunication between a client and one or more of the server-responders.

BRIEF SUMMARY OF THE INVENTION

[0005] The invention provides a method and system capable of ensuring that each client can consistently communicate with one or more servers using delta caching. Building templates for objects at the server(s) is functionally separated from encoding objects for delivery to clients. One or more template-builders are logically separated from one or more delta-encoders. Each operates independently to perform its part of delta caching; template-builders build templates, while delta-encoders use those templates to encode objects for delta caching.

[0006] In an aspect of the invention, template-builders also operate so that those clients not configured for explicit delta caching can perform implicit ("clientless") delta caching by reference to templates maintained at one or more template-builders. In an aspect of the invention, delta-encoders also operate so that the template information and the delta information for any object can be separately compressed

or sent to clients. In one embodiment, delta-encoders operate with template-builders using a client-server technique.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Figure 1 shows a block diagram of a system including a delta caching service.

[0010] Figure 2 shows a process flow diagram of a method including a delta caching service.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0011] In the description herein, a preferred embodiment of the invention is described, including preferred process steps and data structures. Those skilled in the art would realize, after perusal of this application, that embodiments of the invention might be implemented using a variety of other techniques not specifically described, without undue experimentation or further invention, and that such other techniques would be within the scope and spirit of the invention.

Lexicography

[0012] The following terms relate or refer to aspects of the invention or its embodiments. The general meaning of each of these terms is intended to be illustrative and in no way limiting.

[0013] **client**, **server** — In general, these terms refer to devices or software elements operating in a client/server relationship.

[0014] There is no particular requirement that any particular client or any particular server must be a single hardware device or software module. For example, in some embodiments, the client device or the server device might include multiple devices

operating cooperatively (such as when networked) or might include a portion of one or more devices.

[0015] In one embodiment, the client includes a personal computer, such as a workstation, a laptop, or a handheld computer, having a web browser disposed for requesting web objects from the server. In the same embodiment, the server includes one or more web servers, possibly coupled to a network using an element for distributing requests, each of which is disposed for parsing requests for web objects, and for providing those web objects in response thereto. The web objects might be deemed "static," in which case they are retrieved from a file system, a database, or other storage, or might be deemed "dynamic," in which case they are at least partially generated by the web server in response to information that is possibly time-varying.

[0016] There is no particular requirement that any particular client or any particular server must be a single hardware device or software module. For example, in some embodiments, the client device or the server device might include multiple devices operating cooperatively (such as when networked) or might include a portion of one or more devices.

[0017] **delta caching** — In general, this refers to a technique in which a client obtains information regarding a web object in at least two parts: "template information," which refers to information that might be relatively static and is (in one embodiment) retained at the client, and "delta information," which refers to changes from the template information and is (in one embodiment) provided by the

server in response to a difference between a present value of the web object and the template information.

[0018] **delta encoder** — In general, this refers to an element that computes the delta information, such as a difference between a present value of the web object and the template information. In one embodiment, the delta encoder might compress the delta information for delivery to the client.

[0019] **delta information** — In general, this refers to information to be added to the template information to compose an entire web object.

[0020] **encoding web object for delivery** — In general, this refers to a technique in which template information is identified for the web object, delta information is computed in response to the web object and the template information, and the delta information is formatted (such as in an HTTP response message) for delivery to a requesting client.

[0021] **implicit ("clientless") delta caching** — In general, this refers to a delta caching technique in which delta information is formatted for delivery to a requesting client, and in which the formatted delta information includes program fragments (such as JavaScript) stimulating the client to retrieve the template information if that template information is not already at the client.

[0022] **template, template information** — In general, this refers to information which is relatively static, or information that has not changed since the last request for the same web object, or information retained by the client.

[0023] **template builder** — In general, this refers to an element that computes the template information, such as in response to a change in a web object.

[0024] **web page**, **web object** — In general, this refers to an object available at (or dynamically computed at) the server. A web object might include text, pictures, graphics, animation, video or other motion pictures, sound, program fragments or scripts, or other data. When a web object is associated with a specific URL and is intended for presentation by the client, it might be referred to as a web page.

[0025] The scope and spirit of the invention is not limited to any of these definitions, or to specific examples mentioned therein, but is intended to include the most general concepts embodied by these and other terms.

System Elements

[0026] Figure 1 shows a block diagram of a system including a delta caching service.

[0027] A system 100 includes a client 110, a communication network 120, a server 130, a delta encoder 140, and a template builder 150.

Client

[0028] The client 110 includes a workstation having a processor, program and data memory, and mass storage, and is operated by at least one user 111. In one embodiment, the program includes a web browser, disposed for requesting web objects from the server 130, for receiving web objects from the server 130, and for presenting web objects to the user 111. The program and data memory and mass storage collectively include a client cache 112, in which the client 110 records information from the server 130 for presentation to the user 111.

[0029] The client 110 is coupled to the communication network 120, and is disposed for sending request messages 113 to the server 130. In one embodiment, the request messages 113 are formatted in a known protocol, such as HTTP

(hypertext transfer protocol), SHTTP (secure HTTP), FTP (file transfer protocol), or a variant thereof. The client 110 is also disposed for receiving messages from other elements in the system 100.

Network

[0030] The network 120 includes a communication link capable of delivering information between the client 110 and other elements in the system 100. In one embodiment, the communication network 120 includes an internet. However, in alternative embodiments, the network 120 may include an intranet, a LAN or WAN, a portion of a ATM network or PSTN or other switching network, or in general any elements disposed for delivery of information.

Server

[0031] The server 130 includes at least one web server device 131. Each web server device 131 includes a processor, program and data memory, and mass storage. Each web server device 131 is disposed for receiving request messages 113, for retrieving (or dynamically generating) one or more web objects 132 in response thereto, and for formatting a response message 133 including those web objects 132.

[0032] In one embodiment, the server 130 includes a load balancer 134 and a plurality of the web server devices 131. The load balancer 134 includes a processor, and program and data memory, and is disposed for receiving request messages 113 and for delivering those request messages 113 to individual web server devices 131. This has the effect of distributing the workload of responding to request

messages 113 across more than one web server device 131, so the server 130 can respond with relatively less latency to individual request messages 113.

[0033] The web objects 132 might be retrieved from the mass storage, in which case they are deemed "static," or might be dynamically generated by the server 130 (specifically, by one of the web server devices 131) in response to information that might possibly be time-varying. For example, if the web object 132 specified by one of the request messages 113 includes stock quote information, the server 130 would dynamically generate that web object 132 in response to actual stock quotes (such as retrieved from a separate stock quote server).

Delta Encoder

[0034] The delta encoder 140 includes a processor, program and data memory, and mass storage, collectively including a request interceptor 141, a request forwarder 142, an object comparator 143, and a delta formatter 144.

[0035] The request interceptor 141 is coupled to the network 120, and is disposed for intercepting request messages 113 that are destined for the server 130.

[0036] The request forwarder 142 is coupled to the network 120 and to the request interceptor 141, and is disposed for forwarding the intercepted request messages 113 to the server 130.

[0037] In alternative embodiments, the server 130 might be disposed for sending all response messages 133 to the delta encoder 140, in which case the request interceptor 141 and the request forwarder 142 would not be needed. Accordingly, these elements might be regarded as optional.

[0038] The object comparator 143 is coupled to the network 120, and is disposed for receiving response messages 133 from the server 130. The object comparator 143 compares web objects 132 found in those response messages 133, using a set of template information 157 from the template builder 150, and generates delta information 156.

[0039] In one embodiment, the delta encoder 140, from time to time, sends a request to the template builder 150 for the template information 157, and records that template information 157 in its memory or mass storage. The delta encoder 140 would therefore have the template information 157 readily available for computing the delta information 156. In alternative embodiments, the delta encoder 140 sends a request to the template builder 150 for the template information 157 in response to need for that information by the object comparator 143, and uses the template information 157 relatively immediately. The delta encoder 140 would therefore have no special requirement for recording that template information 157 in its memory or mass storage.

[0040] The delta formatter 144 is coupled to the network 120 and to the object comparator 143, and is disposed for formatting the delta information 156 for delivery to the client 110. The delta formatter 144 generates a response message 133 and sends that response message 133 to the network 120 for delivery to the client 110.

[0041] In one embodiment, the delta formatter 144 compresses the delta information 156 before packaging that information in the response message 133.

Template Builder

[0042] The template builder 150 includes a processor, program and data memory, and mass storage, collectively including an object requestor 151, a template identifier 152, and a template server 153.

[0043] The object requestor 151 operates, from time to time, to request web objects 132 from the server 130. The template builder 150 receives the web objects 132 and records them in its memory or mass storage.

[0044] The template identifier 152 operates, from time to time, on web objects 132 recorded in memory or mass storage, and generates template information 157. In one embodiment, the template information 157 can be compressed by the template builder 150 for delivery to the delta encoder 140 or to the client 110.

[0045] The template server 153 operates in like manner as the server 130, in that it receives request messages 113 and sends response messages 133. However, the template server 153 provides template information 157 instead of web objects 132 in response to the request messages 113.

[0046] The template server 153 is available for responding to requests by the delta encoder 140, so the delta encoder 140 can obtain template information with which to compute delta information 156.

[0047] In alternative embodiments, the delta encoder 140 can operate to format the delta information 156 using program fragments such as JavaScript. In this mode of operation, the client 110 does not need to know that the delta information 156 does not include the entire web object 132. Rather, when the client 110 attempts to present the web object 132 (it has only the delta information 156), the program

fragments direct it to request and receive template information 157 from the template server 153, in like manner as it would request and receive a web object 132 from the server 130. Accordingly, the client 110 can obtain both the delta information 156 and the template information 157, and it can present the entire web object 132.

Method of Operation

[0048] Figure 2 shows a process flow diagram of a method including a delta caching service.

[0049] A method 200 is performed by the system 100. Although the method 200 is described serially, the flow points and steps of the method 200 can be performed by separate elements in conjunction or in parallel, whether asynchronously or synchronously, in a pipelined manner, or otherwise. There is no particular requirement that the method 200 must be performed in the same order in which this description lists flow points or steps, except where explicitly so indicated.

Client

[0050] At a flow point 210, the client 110 is ready to request a web object 132 from the server 130.

[0051] At a step 211, the client 110 sends a request message 113 to the server 130.

[0052] At a step 212, the client 110 receives a response message 133 from the delta encoder 140, including delta information 156. In one embodiment, the client 110 recognizes the response message 133 as including only delta information 156, and the client 110 knows it must combine the delta information 156 with template information 157 to present the entire web object 132. In alternative embodiments

such as "clientless" delta caching, the response message 133 includes at least one program fragment directing the client 110 to obtain the template information 157 (either from its client cache 111 or from the template server 153).

[0053] At a step 213, the client 110 determines if it has the requisite template information 157 to present the entire web object 132. If not, the client 110 proceeds with the step 214. If so, the client 110 proceeds with the step 215.

[0054] At a step 214, the client 110 requests the requisite template information 157 from the template server 153 or the delta encoder 140. Template information 157 can be served from both the template server 153 or the delta encoder 140 because they both include the template. A client 110 can be instructed to get the template information 157 from either source. Serving the template information 157 from the template server 153 is particularly advantageous in configurations in which the server 130 is not associated with a delta encoder 140.

[0055] In other embodiments, a CDN (content delivery network) can be used to serve templates information 157. Thus, there are four possibilities: 1) the template builder 150 serves template information 157 directly to the client 110, (2) the delta encoder 140 serves template information 157 directly to the client 110, (3) the template builder 150 serves template information 157 to the client through a CDN and (4) the template builder 150 serves template information 157 to the client 110 through a CDN and the delta encoder 140 serves content to a CDN, which serves it to the client 110

[0056] At a step 215, the client 110 combines the delta information 156 with template information 157 to present the entire web object 132.

[0057] The client 110 returns to the flow point 210.

Server

[0058] The server 130 operates in like manner as an ordinary web server.

Delta Encoder

[0059] At a flow point 220, the delta encoder 140 is ready to encode a web object 132 for delivery to the client 110.

[0060] At a step 221, the object comparator 143 receives the web object 132 from the server 130. As noted above, the delta encoder 140, from time to time, requests template information 157 from the template server 153, and records that template information 157 in its memory or mass storage.

[0061] At a step 222, the object comparator 143 compares the web object 132 with the template information 157, and generates delta information 156.

[0062] At a step 223, the delta formatter 144 formats the delta information 156 for delivery to the client 110, generates a response message 133 and sends that response message 133 to the network 120 for delivery to the client 110. In one embodiment, the delta formatter 144 compresses the delta information 156 before packaging that information in the response message 133.

[0063] The delta encoder 140 returns to the flow point 220.

Template Builder

[0064] At a flow point 230, the template builder 150 is ready to build template information 157. This is done the first time a web object 132 is requested and whenever the delta information 156 associated with that web object 132 grows "too large".

[0065] At a step 231, the object requestor 151 requests one of the web objects 132 from the server 130.

[0066] At a step 232, the template builder 150 receives the one web object 132 and records it in its memory or mass storage.

[0067] At a step 233, the template identifier 152 compares the one web object 132 with an earlier version of that web object 132, and generates template information 157 in response to the comparison.

[0068] At an optional step 234, the template identifier 152 compresses the template information 157.

[0069] The template builder 150 returns to the flow point 230.

[0070] The template server 153 operates in like manner as an ordinary web server, except that it provides template information 157 instead of web objects 132.

Alternative Embodiments

[0071] Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention. These variations would become clear to those skilled in the art after perusal of this application.

[0072] The invention is generally applicable to client-server processes, and to systems in which clients request information and servers provide information in request to responses, such as for example client-server database systems and client-server file systems.

[0073] Those skilled in the art will recognize, after perusal of this application, that these alternative embodiments are illustrative and in no way limiting.